Implementing IDisposable Interface

Jim Fawcett
CSE775 – Distributed Objects
Spring 2005

The following are excerpts from MSDN help pages

The garbage collector automatically releases the memory allocated to a managed object when that object is no longer used, however, it is unpredictable when garbage collection will occur. Furthermore, the garbage collector has no knowledge of unmanaged resources such as window handles, and open files and streams.

Use the Dispose method of this interface to explicitly release unmanaged resources in conjunction with the garbage collector. The consumer of an object can call this method when the object is no longer needed.

It is a version breaking change to add the **IDisposable** interface to an existing class, as it changes the semantics of the class.

For a detailed discussion about how this interface and the Object.Finalize method are used, see the Programming for Garbage Collection and Implementing a Dispose Method topics.

Implementing IDisposable::Dispose()

```
public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}

private void Dispose(bool disposing)
{
    if(!this.disposed)
    {
        if(disposing)
        {
          // call Dispose() on managed resources.
        }
        // clean up unmanaged resources here.
    }
    disposed = true;
}
```

Commented Example - Implementing IDisposable::Dispose()

```
using System;
using System.ComponentModel;

// The following example demonstrates how to create
// a resource class that implements the IDisposable interface
// and the IDisposable.Dispose method.

public class DisposeExample
{
    // A base class that implements IDisposable.
    // By implementing IDisposable, you are announcing that
    // instances of this type allocate scarce resources.
    public class MyResource: IDisposable
    {
        // Pointer to an external unmanaged resource.
        private IntPtr handle;
        // Other managed resource this class uses.
        private Component component = new Component();
        // Track whether Dispose has been called.
        private bool disposed = false;

        // The class constructor.
        public MyResource(IntPtr handle)
        {
            this.handle = handle;
        }

        // Implement IDisposable.
        // Do not make this method virtual.
        // A derived class should not be able to override this method.
        public void Dispose()
        {
            Dispose(true);
            // This object will be cleaned up by the Dispose method.
            // Therefore, you should call GC.SupressFinalize to
            // take this object off the finalization queue
            // and prevent finalization code for this object
            // from executing a second time.
            GC.SuppressFinalize(this);
        }

        // Dispose(bool disposing) executes in two distinct scenarios.
        // If disposing equals true, the method has been called directly
        // or indirectly by a user's code. Managed and unmanaged resources
        // can be disposed.
        // If disposing equals false, the method has been called by the
        // runtime from inside the finalizer and you should not reference
        // other objects. Only unmanaged resources can be disposed.
        private void Dispose(bool disposing)
        {
            // Check to see if Dispose has already been called.
            if(!this.disposed)
            {
                // If disposing equals true, dispose all managed
                // and unmanaged resources.
                if(disposing)
                {
                // Dispose managed resources.
                component.Dispose();
                }
```

```
                // Call the appropriate methods to clean up
                // unmanaged resources here.
                // If disposing is false,
                // only the following code is executed.
                CloseHandle(handle);
                handle = IntPtr.Zero;
            }
            disposed = true;
        }

        // Use interop to call the method necessary
        // to clean up the unmanaged resource.
        [System.Runtime.InteropServices.DllImport("Kernel32")]
        private extern static Boolean CloseHandle(IntPtr handle);

        // Use C# destructor syntax for finalization code.
        // This destructor will run only if the Dispose method
        // does not get called.
        // It gives your base class the opportunity to finalize.
        // Do not provide destructors in types derived from this class.
        ~MyResource()
        {
            // Do not re-create Dispose clean-up code here.
            // Calling Dispose(false) is optimal in terms of
            // readability and maintainability.
            Dispose(false);
        }
    }
    public static void Main()
    {
        // Insert code here to create
        // and use the MyResource object.
    }
}
```

IDisposable Interface

Defines a method to release allocated unmanaged resources.

Classes that Implement IDisposable

AsymmetricAlgorithm
BinaryReader
BinaryWriter
Brush
CacheDependency
Component
ComponentDesigner
Container
Control
CryptoAPITransform
Cursor
CustomLineCap
DesignerTransaction
EncoderParameter
EncoderParameters
EventHandlerList
Font
FontCollection
FontFamily
FromBase64Transform
Graphics
GraphicsPath
GraphicsPathIterator
HashAlgorithm
HttpApplication
Icon
Image
ImageAttributes
IsolatedStorageFile
License
LocalizationExtenderProvider
ManagementObjectCollection
ManagementObjectCollection.ManagementObjectEnumerator
MarshalByValueComponent
Matrix
MessageEnumerator

MessageQueueEnumerator
MessageQueueTransaction
OdbcDataReader
OdbcTransaction
OleDbDataReader
OleDbTransaction
OracleDataReader
OracleTransaction
PaintEventArgs
Pen
Region
RegistryKey
ResourceReader
ResourceSet
ResourceWriter
ResXResourceReader
ResXResourceWriter
SearchResultCollection
ServicedComponent
Socket
SqlCeCommand
SqlCeConnection
SqlCeDataReader
SqlCeEngine
SqlCeRemoteDataAccess
SqlCeReplication
SqlCeTransaction
SqlDataReader
SqlTransaction
Stream
StringFormat
SymmetricAlgorithm
TcpClient
TempFileCollection
TemplateEditingVerb
TextReader
TextWriter
Timer
ToBase64Transform
TraceListener
UdpClient
WaitHandle
WebResponse